In the Claims

1. (Currently amended) A method comprising:

mapping a plurality of exception masks in a source architecture to an exception mask in a target architecture;

determining a translated code block to use as translated code based on the exception mask in the target architecture;

executing the translated code in the target architecture, the translated code representing binary code in the source architecture; and

determining a state for the source architecture if an exception is raised while executing the translated code.

2. (Original) The method of claim 1, wherein determining a state comprises:

determining an excepted instruction in the source architecture corresponding to the translated code that raised the exception;

restoring the state of the source architecture to a pre-instruction state;

masking all exceptions in the exception mask in the target architecture;

re-executing the translated code corresponding to the excepted instruction; and

analyzing a result of re-executing the translated code.

3. (Original) The method of claim 2, wherein analyzing a result comprises:

determining an exception type for the exception that was raised; and

examining the exception mask in the source architecture associated with the excepted instruction.

4. (Original) The method of claim 1 further comprising:

selecting the state for the source architecture based on an exception type if the exception is genuine.

5. (Original) The method of claim 1 further comprising:

selecting a post-instruction state for the source architecture if the exception is erroneous.

6. (Original) The method of claim 1, wherein mapping a plurality of exception masks comprises:

performing a logical AND operation on the plurality of exception masks.

7. (Cancelled)

8. (Currently amended) The method of claim ~~7~~ 1 further comprising:

checking the exception mask against a masking assumption for the translated code block.

9. (Original) The method of claim 8 further comprising:

generating two translated code blocks, an optimized code block with a masking assumption that all exceptions are masked and a conservative code block with a masking assumption that an exception is unmasked.

10. (Original) The method of claim 7, wherein determining a translated code block comprises:

recognizing when the exception mask in the target architecture is changed by an instruction in the source architecture.

11. (Currently amended) A machine-readable medium providing instructions, which when executed by a processing unit, causes the processing unit to perform operations comprising:

mapping a plurality of exception masks in a source architecture to an exception mask in a target architecture;

determining a translated code block to use as translated code based on the exception mask in the target architecture;

executing the translated code in the target architecture, the translated code representing binary code in the source architecture; and

determining a state for the source architecture if an exception is raised while executing the translated code.

12. (Original) The machine-readable medium of claim 11, wherein determining a state comprises:

determining an excepted instruction in the source architecture corresponding to the translated code that raised the exception;

restoring the state of the source architecture to a pre-instruction state;

masking all exceptions in the exception mask in the target architecture;

re-executing the translated code corresponding to the excepted instruction; and

analyzing a result of re-executing the translated code.

13. (Original) The machine-readable medium of claim 12, wherein analyzing a result further comprises:

determining an exception type for the exception that was raised; and

examining the exception mask in the source architecture associated with the excepted instruction.

14. (Original) The machine-readable medium of claim 11 further comprising:

selecting the state for the source architecture based on an exception type if the exception is genuine.

15. (Original) The machine-readable medium of claim 11 further comprising:

selecting a post-instruction state for the source architecture if the exception is erroneous.

16. (Original) The machine-readable medium of claim 11, wherein mapping a plurality of exception masks comprises:

performing a logical AND operation on the plurality of exception masks.

17. (Cancelled)

18. (Currently amended) The machine-readable medium of claim ~~17~~ 11 further comprising:

checking the exception mask against a masking assumption for the translated code block.

19. (Original) The machine-readable medium of claim 18 further comprising:

generating two translated code blocks, an optimized code block with a masking assumption that all exceptions are masked and a conservative code block with a masking assumption that an exception is unmasked.

20. (Original) The machine-readable medium of claim 17, wherein determining a translated code block comprises:

recognizing when the exception mask in the target architecture is changed by an instruction in the source architecture.

21. (Currently amended) An apparatus comprising:

a processing unit coupled to a memory through a bus; and

a binary translation process executed from the memory by the processing unit to cause the processing unit to map a plurality of exception masks in a source architecture to an exception mask associated with the processing unit, determine a translated code block to use as translated code based on the exception mask in the target architecture; execute the translated code representing binary code in the source architecture, and determine a state for the source architecture if an exception is raised while executing the translated code.

22. (Original) The apparatus of claim 21, wherein the binary translation process further causes the processing unit to determine an excepted instruction in the source architecture corresponding to the translated code that raised the exception, restore the state of the

source architecture to a pre-instruction state, mask all exceptions in the exception mask associated with the processing unit, re-execute the translated code corresponding to the excepted instruction, and analyze a result of re-executing the translated code to determine a state for the source architecture.

23. (Currently amended) The apparatus of claim ~~14~~ 22, wherein the binary translation process further causes the processing unit to determine an exception type for the exception that was raised, and examine the exception mask in the source architecture associated with the excepted instruction to analyze a result.

24. (Original) The apparatus of claim 21, wherein the binary translation process further causes the processing unit to select the state for the source architecture based on an exception type if the exception is genuine.

25. (Original) The apparatus of claim 21, wherein the binary translation process further causes the processing unit to select a post-instruction state for the source architecture if the exception is erroneous.

26. (Original) The apparatus of claim 21, wherein the binary translation process further causes the processing unit to perform a logical AND operation on the plurality of exception masks to map the plurality of exception masks to the exception mask associated with the processing unit.

27. (Cancelled)

28. (Currently amended) The apparatus of claim ~~27~~21, wherein the binary translation process further causes the processing unit to check the exception mask against a masking assumption for the translated code block.

29. (Original) The apparatus of claim 28, wherein the binary translation process further causes the processing unit to generate two translated code blocks, an optimized code

block with a masking assumption that all exceptions are masked and a conservative code block with a masking assumption that an exception is unmasked.

30. (Original) The apparatus of claim 27, wherein the binary translation process further causes the processing unit to recognize when the exception mask associated with the processing unit is changed by an instruction in the source architecture to determine the translated code block.